BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) UK Patent Application (19) GB (11) 2 331 387 (13) A

(43) Date of A Publication 19.05.1999

(21) Application No 9724076.6

(22) Date of Filing 15.11.1997

(51) INT CL[6]
G06F 3/12 // G06F 11/00

(52) UK CL (Edition Q)
G4H HPC HQE H1A H13D H14D
G4A AEX AKB1 A12C A12N
U1S S2247

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition P) G4A AKB1 , G4H HPC
INT CL[6] G06F

(71) Applicant(s)
IBM United Kingdom Limited
(Incorporated in the United Kingdom)
PO Box 41, North Harbour, PORTSMOUTH,
Hampshire, United Kingdom

(72) Inventor(s)
Alexander John Wilson

(74) Agent and/or Address for Service
J D Williams
IBM United Kingdom Limited, Intellectual Property
Department, Mail Point 110, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(54) Abstract Title
Controlling printers

(57) A printer controller for a server computer system 90 of a data processing system comprising a user terminal, a printer, and a network 10 interconnecting the server, the user terminal and the printer, receives print jobs from the user terminal via the network, stores the print jobs in a print queue, sequentially sends the print jobs from the print queue to the printer via the network, and automatically deletes from the print queue print jobs stored in the print queue for a period in excess of a predetermined threshold. The jobs are accompanied by print drivers, and the server detects when the driver does not match the printer and sends an alert or replaces the driver.

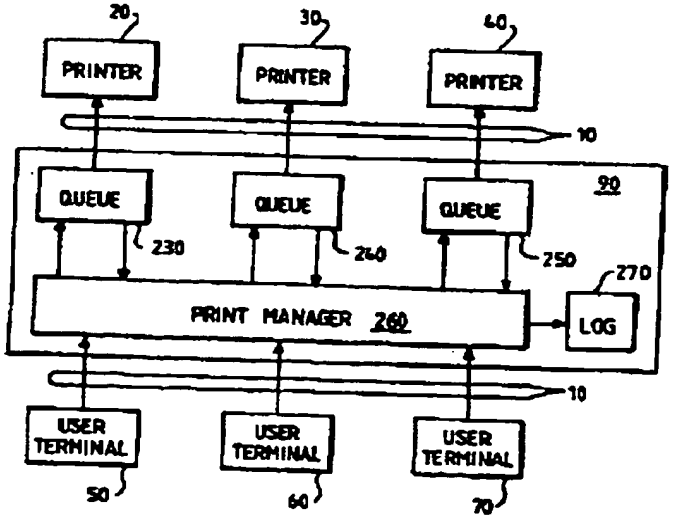


FIG. 3

GB 2 331 387 A

2331387

METHOD AND APPARATUS FOR CONTROLLING PRINTERS

The present invention relates to a method and apparatus for controlling printers in a distributed data processing environment.

In a typical distributed data processing environment, there are multiple interconnected local area networks (LANs) each having a plurality of connected user terminals. The interconnected LANs may be spread throughout a building or site to form a campus network portion of the distributed data processing environment. Such a data processing environment may comprise multiple campus networks interconnected to form a wide area network (WAN). In general, each LAN in the data processing environment comprises a server computer system for providing application services to user terminals on request. Examples of such applications include word processor and computer aided drawing software packages. A plurality of printers are typically connected to each LAN for printing files produced by connected users via the word-processing and drawing applications. The printers are controlled by print manager software executing either on the server computer system providing the application services or, more commonly, on a separate print server computer system also connected to the LAN. The print manager has a separate queue for each printer. Any jobs sent by users to a printer in the LAN are stored in the corresponding queue prior to being sent sequentially to the printer.

In a typical distributed data processing environment, each LAN is furnished with a range of different printers each optimised for different type of printing. Generally, each printer has it's own configuration or driver software which is down-loaded to the printer via the print manager along with each print job from the user community. When a user sends a job to a queue using the wrong driver, the queue for the specified printer becomes jammed because the printer is unable to accept the job. This problem exasperated by printer driver names changing between different operating systems on the user terminals. Users may be unaware of such differences and unwittingly select the wrong drivers. Jammed print queues lead to delays in printing throughput.

Another problem with conventional print managers is that they provide no guidance to assist in planning the positioning of printers within a building. Under-utilisation of printers may stem from in convenient positioning.

Furthermore, it would be desirable to remotely monitor print queues over a WAN and to have alerts from printers routed to a central monitoring point thereby enabling operators to identify printing problems centrally. Local site support can then be contacted with a view to resolving such problems.

If a large number of print jobs are allowed to build up in a print server, because a number of printers are off-line perhaps, then spooler directories in the print server may overflow thereby crashing the server.

In accordance with the present invention, there is now provided printer control apparatus for a server computer system of a data processing system comprising a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the apparatus comprising a receiver for receiving print jobs from the user terminal via the network, storage means for storing the print jobs in a print queue, a transmitter for sequentially sending the print jobs from the print queue to the printer via the network, and purging means for automatically deleting from the print queue print jobs stored in the print queue for a period in excess of a predetermined threshold.

Preferably, the storage means further comprises a print log for recording each print job received from the user terminal.

Examples of the present invention may further comprise means for generating an alert message on detection of a mismatch between a print job stored in the print queue and a print driver associated with the print job.

Preferred embodiments of the present invention may further comprise means for sending the alert message to the user terminal via the network. Alternatively, in some embodiments of the present invention, there may provided means for sending the alert message to a network management terminal via the network. Also, in some embodiment of the present invention, there may be provided means for correcting the mismatch by automatically replacing the printer driver associated with the print job stored in the print queue.

It will be appreciated that the present invention extends to a server computer system comprising a central processing unit, a memory, printer control apparatus as hereinbefore described, and a bus

architecture interconnecting the central processing unit, the memory, and the printer control apparatus. It will also be appreciated that the present invention further extends to a data processing system comprising a user terminal, a printer, a server computer system as hereinbefore described, and a network interconnecting the user terminal, the printer, and the server computer system.

Viewing the present invention form another aspect, there is now provided a method for controlling a printer in a data processing system comprising a server computer system, a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the method comprising: receiving at the server print jobs from the user terminal via the network; storing the print jobs in a print queue in the server, sequentially sending the print jobs from the print queue in the server to the printer via the network; and, automatically deleting from the print queue in the server print jobs stored in the print queue for a period in excess of a predetermined threshold.

Viewing the present invention from yet another aspect, there is now provided a computer program product for a server computer system of a data processing system comprising a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the product comprising first code means for receiving print jobs from the user terminal via the network, second code means for storing the print jobs in a print queue, third code means for sequentially sending the print jobs from the print queue to the printer via the network, and fourth code means for automatically deleting from the print queue print jobs stored in the print queue for a period in excess of a predetermined threshold.

Preferred embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a distributed data processing environment;

Figure 1 is a block diagram of a computer system of the data processing environment;

Figure 3, is a block diagram of a print server of the data processing environment;

Figure 4 is a block diagram of the print server in combination with a user terminal and a printer of the data processing environment; and,

Figure 5 is a flow diagram of a print manager portion of the print server.

Referring first to Figure 1 a distributed data processing system comprises a network 10. A plurality of user terminals 50-70 are connected to the network 10. A plurality of printers 20-40 are also connected to the network 10. Also connected to the network 10 are an application server computer system 80 and a print server computer system 90.

Referring to Figure 2, the printer server 90, application server 80, and the user terminals 50-70, each comprise a computer system having a random access memory (RAM) 110, a read only memory (ROM) 120, a central processing unit (CPU) 100, a mass storage device 130 comprising one or more large capacity magnetic disks or similar data recording media, a network adaptor 180, a keyboard adaptor 170, a pointing device adaptor 160, and a display adaptor 150 all interconnected via a bus architecture 140. A keyboard 200 is coupled to the bus architecture 140 via the keyboard adaptor 170. Similarly, a pointing device 210, such as a mouse, touch screen, tablet, tracker ball or the like, is coupled to the bus architecture 140 via the pointing device adaptor 160. Equally, a display output device 220, such as a cathode ray tube (CRT) display, liquid crystal display (LCD) panel, or the like, is coupled to the bus architecture 140 via the display adaptor 150. The bus architecture 140 is additionally coupled to the network 10 via the network adapter 180.

Basic input output system (BIOS) software is stored in the ROM 120 for enabling data communications between the CPU 100, mass storage 130, RAM 110, ROM 120, and the adaptors 150-180 via the bus architecture 140. Stored on the mass storage device 130 is operating system software and application software. The operating system software cooperates with the BIOS software in permitting control of the computer system by the application software.

In the mass storage of the application server 80 there are stored a plurality of application software packages such as word-proc ssing packages and drawings packages for use by the user terminals 50-70.

Referring to Figure 3, the printer server 90 comprises a printer manager 260 for controlling supply of printing jobs from the user terminals 50-70 connected to the network 10 to the printers 20-40 also connected to the network 10. It will be appreciated that the print manager 260 may be stored in the mass storage 130 of the print server 90 for execution by the CPU 100 of the print server 90. The mass storage 130 of the print server 90 also contains a plurality of print queues 230-250 each corresponding to a different one of the printers 20-40. A print log 270 is also maintained in the mass storage 130 of the print server 90.

With reference now to Figure 4, suppose for example that a user of user terminal 50 wishes to print, on printer 20, a document prepared on a word processor application loaded into user terminal 50 from the application server 80. The user specifies the desired printer, printer 20, when issuing the print request. The document is sent from the user terminal 50 to the print server via the network 20 as a print job 280 with an accompanying print driver 290 corresponding to specified printer 20. At the print server 90, the print job 280 and driver 290 are temporarily stored in the queue associated with printer 20 until jobs already in the queue 230 are completed by printer 20. When the print job 280 reaches the head of the queue 230, it is sent to printer 20 for printing.

As mentioned earlier, the print manager 260 may be stored in the mass storage 130 of the print server 90 for execution by the CPU 100 of the print server 90. In a particularly preferred embodiment of the present invention, the print manager 260 is written in the C computer programming langauge. n operation, the printer manager 260 examines all incoming print and checks that they are using the correct driver for the intended printer 20-40. In the aforementioned particularly preferred embodiment, the print manager 260 comprises a REXX cmd file which is called by the C program when a problem is detected prompting an alert. The cmd file can be configured to send alerts to a central point using a proprietary protocol. The cmd file also deletes any print jobs remaining in a queue 230-250 beyond a predetermined time. Furthermore, the cmd file can be configured to send alerts to users and/or administrators in the event that an incorrect printer driver is specified.

In operation, the print manager 260 constantly monitors print queues 20-40 on the print server 90. Thus, the print manager can examine all jobs coming into each queue 230-250. In the aforementioned particularly preferred embodiment, the print manager 260 is configured by altering an INI file which determines the actions for the print manager 260 to take. Specifically, the INI file can be configured to: match a driver to a queue holding a corresponding job in the event that a mismatch is detected; send alerts relating to failing queues, jobs, and devices; delete any jobs in a queue that are over a specified age; and, log all print jobs in the print log 270.

What follows is a pseudo-code listing of the INI file showing the aforementioned functions provided by the print manager 260.

```
;PRINTMAN.INI
;This is the ini file that is called at initialisation of PRTMAN.EXE.
;This is the driver matching section
;MATCH: parameters:- yes or no
;This can be either yes or no. If yes it will match the JOB DRIVER to the
DEVICEDRIVERS.
;If the JOB DRIVER is unsupported the job will be held.
[DRIVERS]
match = yes

;This is the logging section
;log: parameters:- yes or no if yes all jobs are logged otherwise not
;use VIEWI.OG.EXE to view the file, You need to stop printman by pressing
Q
;to do this
[LOGGING]
log = yes

;This is the NOTIFY section
;alertusers: parameters yes or no if yes users are sent network messages
;otherwise not ;alertadmin: parameters
;yes or no if yes administrators are alerted otherwise not
;adminid: parameters id of an administrator use this to indicate an admin
id
;you wish to alert
[NOTIFY]
alertuser - yes
```

```
alertadmin = yes
adminid = alexl :

;This is the job count section. Enter the no of jobs to allow in a queue
;before sending an alert ;with the jobcount parameter. The timing of the
alert
;is altered with the jfreq parameter below.
;20 is the default forjobcount.
[NOJOBS)
jobcount = 20

;This is the alerting section.
;Parameters: yes or no
;Choosing yes enables the program to call PRTCMD.cmd a rexx cmd file that
can ;be used for example to send alerts remotely.
;queuealerts:-yes or no to turn on/off queuealerts
;qfreq:- no of queuealerts to ignore before alert is sent 4000 is the
default
;devicealerts:-yes or no to turn on/off devicealerts
;dfreq:-no of devicealerts to ignore before alert is sent 4000 is the
default
;jobalerts:-yes or no to turn on/off jobalerts
;jfreq:-no ofjobalerts to ignore before alert is sent 4000 is the default
[ALERTING]
enable - yes
queuealerts= yes
qfreq = 8000
devicealerts= yes
dfreq = 8000
jobalerts= yes
jfreq = 8000

;This is the scavenge section.
;If run is yes it will delete all jobs found on a print server that are
older ;than the specified kill time in hours. ktime will default to 12
hours if not ;specified & run = yes
[SCAVENGE]
run = yes
ktime =17
```

Referring now to Figure 5, the print manager 260 is initially configured, at 300, via the aforementioned INI file. The print manager 260 operates by enumerating, at 310, all the print queues 230-250 on the print server 310. Then, the print manager 260 checks, at 320, the printers 20-40 attached to each queue 230-250. The print manager 260 follows the printer check 320 with a check, at 330, of all jobs waiting in the queues 230-250. In the event of a problem in a queue 230-250, a printer 20-40, or a job, a corresponding message is sent to a circular queue 340 in the print manager 260. The circular queue 340 is monitored by a separate thread 350 of the print manager 260. On detection of a message, the thread 350 calls PRTMAN.CMD, a rexx command file which, in turn, is sent the message as a parameter. PRTMAN.CMD can be configured via rexx to deal with the message. For example, PRTMAN.CMD may be configured to incorporate the message in a network message to a central administrator or a user via any proprietary protocol.

Referring back to Figure 3, in addition to fault reporting, the print manager 260 logs all jobs received in a log file 270. In particularly preferred embodiments of the present invention, the log file 270 is cyclical, holding of the order of 100000 jobs before recycling, or however many jobs are deemed sufficient to accommodate a year of average printing.

Returning to Figure 5, if the print manager detects, at 330, that any waiting job is over a predetermined age (eg: 12 hours, set in the INI file), the print manager 260 automatically deletes the job from the holding queue 230-250. The print manager 260 thus keeps the queue 230-250 clear of over-age jobs. In especially preferred embodiments of the present invention, the age threshold for purging jobs from the print queues 230-250 is adjustable by network administrators according to prevailing demands on the printers 20-40. Also, in especially preferred embodiments, the frequency with which queues are checked by the print manager 260 (eg: every 10 ms) is adjustable by network administration staff, again ,according to prevailing demands on the printers 20-40.

It will be appreciated that the printer manager 260 hereinbefore described prevents print queues from jamming due to users specifying incorrect printer drivers. It will also be appreciated that the print manager 260 also prevents print queues 20-40 from overfilling with jobs by automatically deleting over-age jobs. Furthermore, it will be appreciated that, by logging incoming print jobs, the print manager 260

assists with determining the usage and location planning of printer installations. Still furthermore, the alerts generated by the print manager 260 enable printer problems to be quickly notified to a local or central administration points.

Preferred embodiments of the present invention have been hereinbefore described with reference to a data processing system in which the print server 260 is implemented in a separate computer system to the application server 80. However, it will be appreciated that the present invention is equally applicable to data processing systems in which the print server 90 and the application server 80 are integrated in the single computer.

In summary what has been hereinbefore described by way of example of the present invention is a printer controller 260 for a server computer system 90 of a data processing system comprising a user terminal 50-70, a printer 20-40, and a network 10 interconnecting the server 90, the user terminal 50-70 and the printer 20-40, which printer controller 260 receives print jobs from the user terminal 50-70 via the network 10, stores the print jobs in a print queue 230-250, sequentially sends the print jobs from the print queue 230-250 to the printer 20-40 via the network 10, and automatically deletes from the print queue 250-270 print jobs stored in the print queue 250-270 for a period in excess of a predetermined threshold.

CLAIMS

1. Printer control apparatus for a server computer system of a data processing system comprising a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the apparatus comprising a receiver for receiving print jobs from the user terminal via the network, storage means for storing the print jobs in a print queue, a transmitter for sequentially sending the print jobs from the print queue to the printer via the network, and purging means for automatically deleting from the print queue print jobs stored in the print queue for a period in excess of a predetermined threshold.

2. Apparatus as claimed in claim 1, wherein the storage means further comprises a print log for recording each print job received from the user terminal.

3. Apparatus as claimed in claim 1 or claim 2, comprising means for generating an alert message on detection of a mismatch between a print job stored in the print queue and a print driver associated with the print job.

4. Apparatus as claimed in claim 3, comprising means for sending the alert message to the user terminal via the network..

5. Apparatus as claimed in claim 3, comprising means for sending the alert message to a network management terminal via the network.

6. Apparatus as claimed in claim 3 comprising means for correcting the mismatch by automatically replacing the printer driver associated with the print job stored in the print queue.

7. A server computer system comprising a central processing unit, a memory, printer control apparatus as claimed in any preceding claim, and a bus architecture interconnecting the central processing unit, the memory, and the printer control apparatus.

8. A data processing system comprising a user terminal, a printer, a server computer system as claimed in claim 7, and a network interconnecting the user terminal, the printer, and the server computer system.

9. A method for controlling a printer in a data processing system comprising a server computer system, a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the method comprising: receiving at the server print jobs from the user terminal via the network; storing the print jobs in a print queue in the server, sequentially sending the print jobs from the print queue in the server to the printer via the network; and, automatically deleting from the print queue in the server print jobs stored in the print queue for a period in excess of a predetermined threshold.

10. A computer program product for a server computer system of a data processing system comprising a user terminal, a printer, and a network interconnecting the server, the user terminal and the printer, the product comprising first code means for receiving print jobs from the user terminal via the network, second code means for storing the print jobs in a print queue, third code means for sequentially sending the print jobs from the print queue to the printer via the network, and fourth code means for automatically deleting from the print queue print jobs stored in the print queue for a period in excess of a predetermined threshold.

The Patent Office

12.

Application No: GB 9724076.6
Claims searched: 1-10

Examiner: Mike Davis
Date of search: 12 February 1998

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.P): G4H (HPC), G4A (AKB1)

Int Cl (Ed.6): G06F

Other:

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
	None	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

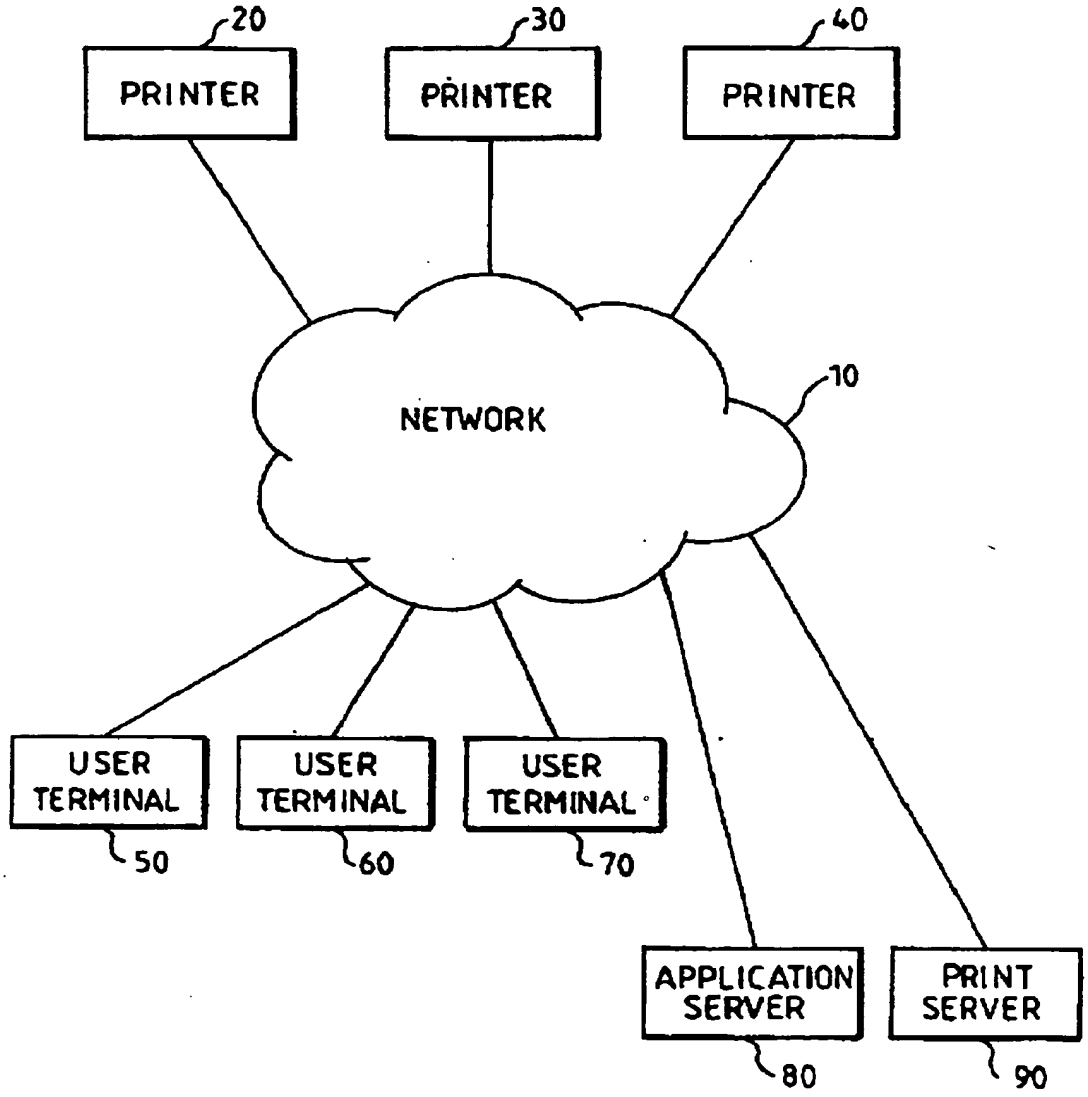


FIG. 1

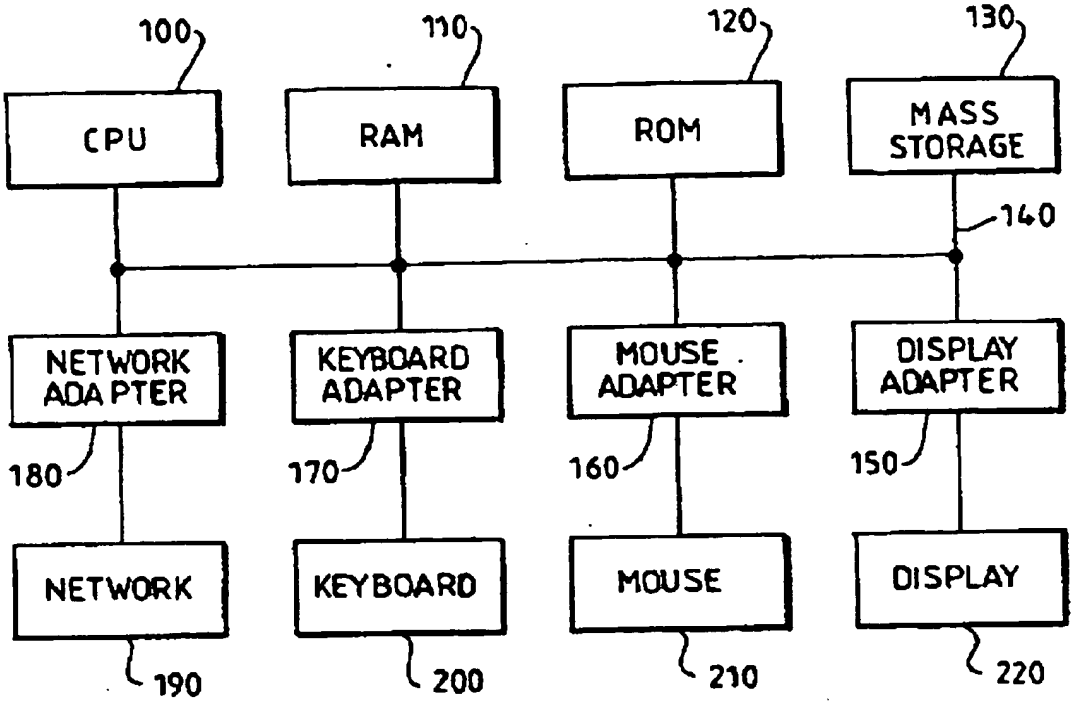

100
110
120
130
CPU
RAM
ROM
MASS STORAGE
140
NETWORK ADAPTER
KEYBOARD ADAPTER
MOUSE ADAPTER
DISPLAY ADAPTER
180
170
160
150
NETWORK
KEYBOARD
MOUSE
DISPLAY
190
200
210
220

FIG. 2

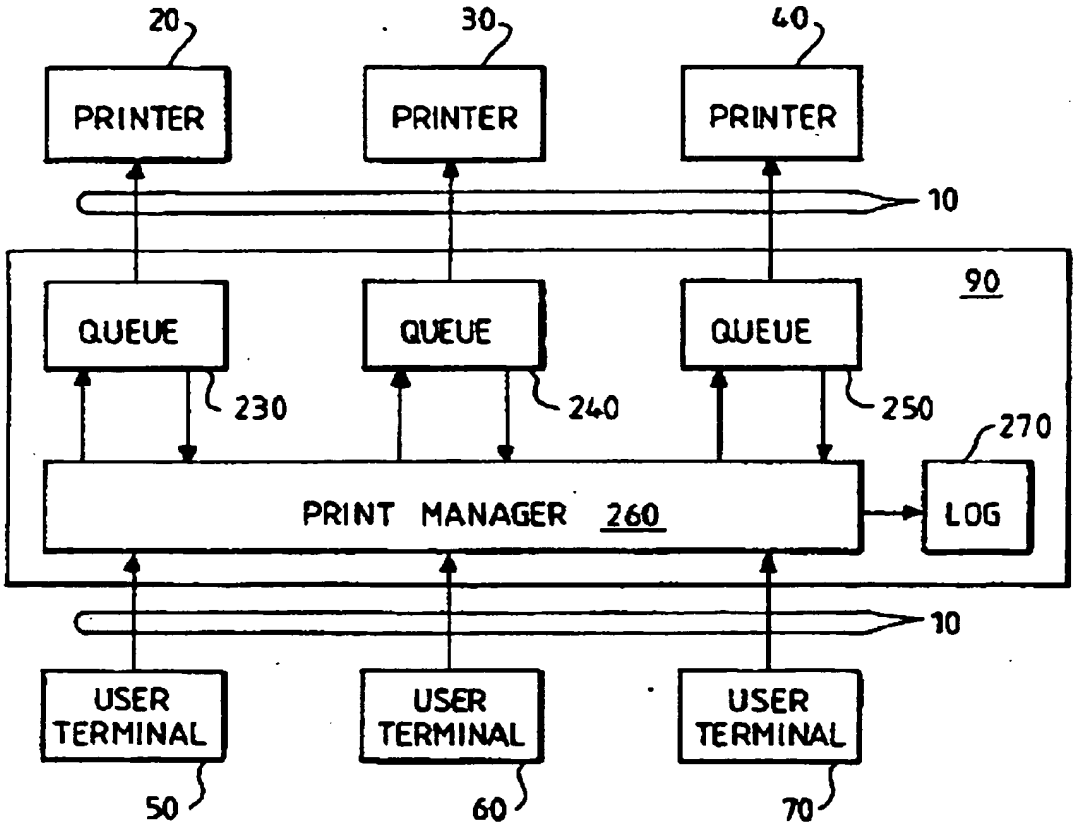


FIG. 3

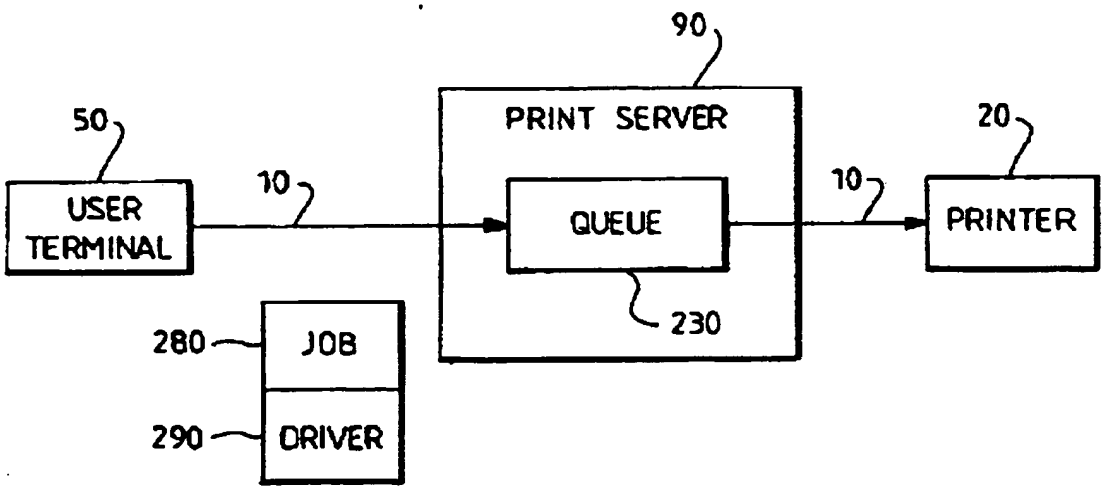

90
PRINT SERVER
50
USER TERMINAL
10
QUEUE
230
10
20
PRINTER
280
JOB
290
DRIVER

FIG. 4

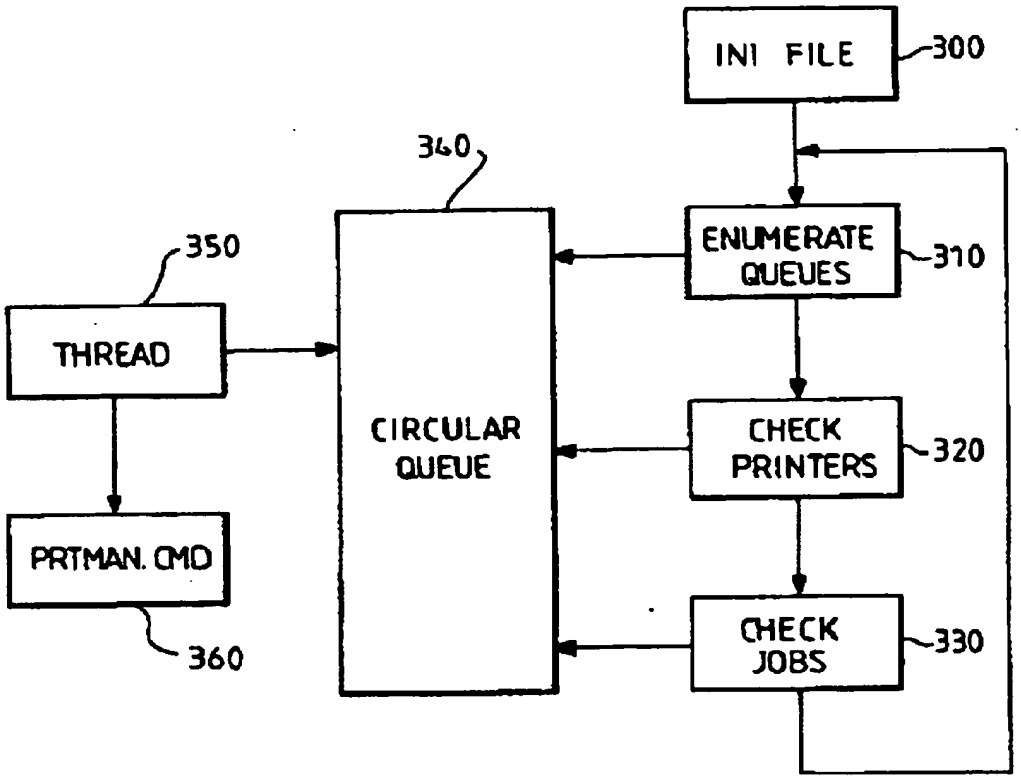


FIG. 5

HP Docket No. 10990088

U.S. Patent Application S/N: 09/640,855
Filed: August 17, 2000
Inventor: William Hertling

SYSTEM AND METHOD FOR NETWORK PRINTING USING A PEER HYBRID PRINTING PROTOCOL

TECHNICAL FIELD

The present invention is generally related to the field of network printing and, more particularly, is related to a system and method for network printing using a peer hybrid printing protocol.

BACKGROUND OF THE INVENTION

In printing documents from client computers and other devices to network printers, various printing protocols are used. Three such typical printing protocols employed include client-server printing, server-side-rendering (SSR) client-server printing, and peer-to-peer printing. These protocols are used by the various devices on the network to cause the printing of a particular document from the client. Such devices may include, for example, one or more clients, a queue server, a print server, and a printer, *etc.*

To print a document using client-server printing, for example, an application in a client first makes graphical device interface (GDI) calls to the operating system in the client. The operating system then creates enhanced meta file data (EMF data) from the GDI calls that are handed to a printer driver. The printer driver then renders the EMF data into printer ready bits (PRB) that form a print job. The print job is applied to the operating system to be transmitted over the network to the queue server for printing.

The queue server then receives the print job and places it in a queue associated with a particular printer. When the printer is ready to receive the print job, the queue server transmits the print job over the network to the print server. The print server then applies the print job to the printer for printing. Unfortunately, the client-server approach requires the entire document to be transmitted on the network twice, thereby consuming a potentially large amount of network bandwidth.

In server-side-rendering, (SSR) client-server printing, an application in a client first makes GDI calls to the operating system in the client. The operating system then creates EMF data from the GDI calls. The EMF data is transmitted over the network to a queue server. In the queue server, the EMF data is placed in a queue for printing. When the

document is to be printed, the corresponding EMF data is applied to a printer driver in the queue server that renders the EMF data into printer ready bits (PRB) that form a print job. When the printer is ready to receive the print job, the print job is applied to the operating system in the queue server to be transmitted over the network to the print server for printing. The print server then applies the print job to the printer for printing. Unfortunately, the SSR client-server approach also requires the entire document to be transmitted on the network twice, once as EMF data and once as printer ready bits. This results in the consumption of a potentially large amount of network bandwidth. In addition, the centralization of the printer driver on the queue server can stress the computing capacity of the queue server to effectively render the EMF data into printer ready bits (PRB). This is especially a concern given that a number of printer drivers may be placed on the queue server to service a number of printers.

In peer-to-peer printing, an application in a client first makes GDI calls to the operating system in the client. The operating system then creates EMF data from the GDI calls that are handed to a printer driver in the client. The printer driver then renders the EMF data into printer ready bits (PRB) that form a print job. The print job is applied to the operating system in the client to be transmitted over the network directly to the print server for printing.

When the print server is ready to receive data, the operating system in the client transmits the print job to the print server. The print server then applies the print job to the printer for printing. Although, the peer-to-peer printing approach only requires the entire document to be transmitted on the network once, the peer-to-peer approach does not facilitate centralized printer control or queuing.

SUMMARY OF THE INVENTION

In light of the foregoing, the present invention provides for a system and method for network printing of a document. In one embodiment, the network includes a client coupled to the network, the client having a first processor coupled to first local interface and a first memory coupled to the first local interface. The client also includes send job logic stored on the first memory and executable by the first processor. The send job logic includes logic

to generate a print j b ticket associated with a full print job, and logic to transmit the print job ticket to a queue server that is coupled to the network.

The queue server includes a second processor coupled to second local interface and a second memory coupled to the second local interface. The queue server also includes print queue logic stored on the second memory and executable by the second processor. The print queue logic includes logic to place the print job ticket in a queue in the queue server, and logic to transmit the print job ticket from the queue server to a print server that is coupled to the network.

The print server has a third processor coupled to third local interface and a third memory coupled to the third local interface. The print server further includes print server logic stored on the third memory and executable by the third processor. The print server logic comprises logic to determine an address of the client on the network from the print job ticket received from the queue server, logic to transmit a request for a full print job to the client, and logic to apply the full print job received from the client to the printer.

In addition to the above system, further aspects of the present invention include the functionality of each of the client, queue server, and print server related to the printing protocol employed to print a document on a network printer.

The present invention may also be viewed as a method for printing on a network. Broadly stated, the method comprises the steps of: generating a print job ticket in a client, the print job ticket being associated with a full print job; transmitting the print job ticket to a queue server; placing the print job ticket in a queue in the queue server; transmitting the print job ticket from the queue server to a print server; transmitting a request for a full print job from the print server to the client; and, transmitting the full print job from the client to the print server in response to the request.

In addition to the above method, further aspects of the present invention include the steps taken with reference to the individual client, queue server, and print server related to the printing protocol employed to print a document on a network printer.

The systems and method of the present invention provide several advantages, such as, printing documents on a network printer while using less bandwidth and at the same time retaining centralized control over the printing functions of the network.

Other features and advantages of the present invention will become apparent to a person with ordinary skill in the art in view of the following drawings and detailed description. It is intended that all such additional features and advantages be included herein within the scope of the present invention.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The invention can be understood with reference to the following drawings. The components in the drawings are not necessarily to scale. Also, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a block diagram of a network according to an embodiment of the present invention;

FIG. 2 is a block diagram that depicts a printing protocol implemented on the network of FIG. 1;

FIG. 3 is a flow chart of job send logic executed by a client in the network of FIG. 1 to implement the printing protocol of FIG. 2;

FIG. 4 is a flow chart of print queue logic executed by a queue server in the network of FIG. 1 to implement the printing protocol of FIG. 2; and

FIG. 5 is a flow chart of print server logic executed by a print server in the network of FIG. 1 to implement the printing protocol of FIG. 2.

DETAILED DESCRIPTION OF THE INVENTION

With reference to FIG. 1, shown is a printing network 100 according to an embodiment of the present invention. The printing network 100 includes a network 103 to which is coupled a client 106, a queue server 109, and a print server 113. A printer 116 is coupled to the print server 113 as shown.

The client 106 includes a processor circuit that includes a processor 133 and a memory 136, both of which are coupled to a local interface 139. The local interface 139 may be a data bus with an accompanying control bus as known by those with ordinary skill in the art. The client 106 further includes a network interface 143 that couples the local

interface 139 to the network 103. Through the network interface 143, data may be transmitted from the local interface 139 to the network 103 to any other device on the printing network 100. Also, the network interface 143 makes data transmitted via the network 103 available on the local interface 139. Thus, the network interface 143 may be, for example, a network interface card, modem, or other interface that includes appropriate buffer circuitry and transmission circuitry, *etc.*, to accomplish these tasks. In this respect, the network 103 may comprise a local area network, a wide area network, or the Internet, *etc.*

The client 106 also features an operating system 153 and a printer driver 156 stored in the memory 136 and executable by the processor 133. The operating system 153 is executed to control and/or perform the various functions of the client 106 in a similar manner to operating systems known by those with ordinary skill in the art. However, according to the present invention, the operating system 153 further includes job send logic 159 that is executed in order to interface with the queue server 109 and the print server 113 to print a document as will be discussed. Such a document (not shown) generally results from the operation of a particular application (not shown) on the client 106 as is generally known by those with ordinary skill in the art. The printer driver 156 is employed by the client device 106 to render enhanced meta file (EMF) data into printer ready bits (PRBs) as is known by those with ordinary skill in the art. The particular printer driver 156 that is used is compatible with the printer 116.

The queue server 109 includes a processor circuit that includes a processor 173 and a memory 176, both of which are coupled to a local interface 179. The local interface 179 may be a data bus with an accompanying control bus as known by those with ordinary skill in the art. The queue server 109 further includes a network interface 183 that couples the local interface 179 to the network 103. The network interface 183 is similar to the network interface 143 and is not described in great detail.

The queue server 109 further comprises an operating system 193 that is executed to control and/or perform the various functions of the queue server 109 in a similar manner to operating systems known by those with ordinary skill in the art. However, according to the present invention, the operating system 193 additionally includes print queue logic 196 that is executed to interface with the client 106 and the print server 113 in accomplishing the

tasks of queuing and generally printing a document from the client 106 on the printer 116 as will be discussed.

The print server 113 includes a processor circuit that features a processor 213 and a memory 216, both of which are coupled to a local interface 219. The local interface 219 may be a data bus with an accompanying control bus as known by those with ordinary skill in the art. The print server 113 further includes a network interface 223 that couples the local interface 219 to the network 103. The network interface 223 is similar to the network interface 143 and is not described in great detail.

The print server 113 also includes a printer interface 226 that may be a printer interface card that couples the printer 116 to the local interface 219. In this regard, the printer interface 226 may comprise an appropriate printer card, *etc.*, as known by those with ordinary skill in the art.

The print server 113 further comprises an operating system 233 that is executed to control and/or perform the various functions of the print server 113 in a similar manner to operating systems known by those skilled in the art. However, according to the present invention, the operating system 233 additionally includes print server logic 236. The print server logic 236 is executed to interface with the client 106 and the queue server 109 in accomplishing the tasks of queuing and generally printing a document from the client 106 on the printer 116 as will be discussed.

Additionally, the memories 136, 176, and 216 may include both volatile and nonvolatile memory components. Volatile components are those that do not retain data values upon loss of power. Conversely, nonvolatile components retain data upon a loss of power. Thus, the memories 136, 176, and 216 may comprise, for example, random access memory (RAM), read-only memory (ROM), hard disk drives, floppy disks accessed via an associated floppy disk drive, compact disks accessed via a compact disk drive, magnetic tapes accessed via an appropriate tape drive, and/or other memory components, or a combination of any two or more of these memory components.

Also, each of the processors 133, 173, 213 may represent multiple processors operating in parallel and each of the memories 136, 176, 216 may represent multiple memories. In such a case, each of the local interfaces 139, 179, 219 may be an appropriate network that facilitates communication between any two of the multiple processors or

between any one processor and any one of the memories, *etc.* In addition, the processors 133, 173, 213, memories 136, 176, and 216, and local interfaces 139, 179, 219 may be electrical or optical in nature. The memories 136, 176, and 216 may also be magnetic in nature.

The client 106, the queue server 109, and the print server 113 may include appropriate input/output devices (not shown). In this regard, the input devices may include, for example, but are not limited to a keyboard, keypad, touch pad, touch screen, microphone, mouse, joystick, or one or more push buttons, *etc.* User output devices may include display devices, indicator lights, speakers, printers, *etc.* Such display devices may be, for example, cathode ray tubes (CRTs), liquid crystal display screens, gas plasma-based flat panel displays, indicator lights, light emitting diodes, and other display devices.

In addition, the queue server 109 and the print server 113 may also be combined into a single server that performs the functions of both, taking the operation requirements of both into account. That is to say, a combined queue and print server should have a fast enough operating speed to accomplish both functions simultaneously.

With to reference to FIG. 2, shown is a block diagram of the printing network 100 that depicts a printing protocol 300 according to an embodiment of the present invention. The printing protocol 300 provides for communication between the various devices of the printing network 100. In particular, the printing protocol 300 includes a print job ticket 303, a printer polling message 304, a printer response message 306, a request for full print job 309, a full print job 313, and a printer idle signal 316. Each of these data communications are employed as part of the printing protocol 300 to cause a document that resides in the client 106 to be printed on the printer 116.

Assuming that the client 106 includes an application that has generated a document to be printed on the printer 116, the client 106 makes appropriate graphical device interface (GDI) calls to the operating system 153 (FIG. 1). The operating system 153 makes corresponding enhanced meta file (EMF) data from the GDI calls, and hands the EMF data to the printer driver 156. The printer driver 156 renders the EMF data into printer ready bits (PRBs) and sends the resulting full print job to the operating system 153 in PRB format. The operating system 153 then spools the full print job to the memory 136.

The client 106 then generates a print job ticket 303 that includes the address of the client 106 on the network 103, the name or identification of the full print job, and any other pertinent information such as the number of pages to be printed, *etc.* The client 106 then transmits the print job ticket 303 to the queue server 109. The queue server 109 places the print job ticket 303 in a printing queue maintained in the queue server 109. The queue server 109 then transmits a printer polling message 304 to the print server 113 to determine if the printer 116 is available to print a document. The print server 113 responds with the printer response message 306 that informs the queue server 109 that the printer 116 is busy printing or is available. If the printer 116 is occupied with another print job, the queue server 109 waits for a period of time and then retransmits the printer polling message 304. If the printer 116 is available, the queue server 109 then transmits the print job ticket 303 to the print server 113.

Alternatively, rather than polling the print server 113 as to the printer availability, the queue server 109 may simply wait for a "printer available signal" from the print server 113 signifying that a previous print job sent to the printer 116 has finished. When the "printer available signal" is received, the queue server 109 is informed that the printer 116 is available for another print job.

Once the print server 113 receives the print job ticket 303, the print server 113 examines it to ascertain the client 106 that originated the print job ticket 303, assuming that there are a number of clients 106 on the network 103. The print server 113 then transmits a "request for full print job" 309 to the client 106 based upon the client network address.

In response, the client 106 transmits the full print job 313 to the print server 113. The full print job 313 includes the digital data that makes up the document to be printed in PRB format. Thereafter, the print server 113 transmits the full print job 313 to the printer 116 that prints the document, accordingly. Once the printer 116 has finished printing the document, the printer 116 transmits the printer idle signal 316 back to the print server 113. The print server 113 then waits for another printer polling message 304 to begin the process again. Alternatively, if printer polling is not used by the queue server 109, the print server 113 transmits a "printer available signal" to the queue server 109 to inform the queue server 109 that the printer is ready for the next print job. The

print server 113 maintains the availability status of the printer 116 based upon the fact that it sent a full print job 313 to the printer 116 and/or received the printer idle signal 316 from the printer 116.

With the foregoing in mind, reference is made to FIG. 3 that shows a flowchart of the job send logic 159 that is executed by the client 106 (FIG. 1). The job send logic 159 is executed by the processor 133 (FIG. 1) in order to fulfill the role or the client 106 in printing a document on the printer 116 (FIG. 1) according to the printing protocol 300 (FIG. 2). Beginning with block 350, the job send logic 159 determines whether a document is to be printed as mandated by the operating system 153 (FIG. 1). If such is the case then the job send logic 169 moves to block 353 in which the print job ticket 303 (FIG. 2) is generated and transmitted to the queue server 109 (FIG. 2). If there is no document to be printed in block 350, then the job send logic 159 moves to block 356 in which it is determined whether a request for full print job 309 (FIG. 2) has been received from the print server 113 (FIG. 2). Also, after the print job ticket 303 is transmitted to the queue server 109 in block 353, the job send logic 159 moves to block 356 as shown. Thus, the job send logic 159 will continually move between blocks 350 and 356 waiting for either a document to be printed or a printer request to be received from the print server 113.

If there is no request for full print job 309 received from the print server 113 in block 356, then the job send logic 159 reverts back to block 350 as shown. On the other hand, if a request for full print job 309 is received by the client 106 in block 356, then the job send logic 159 moves to block 359. In block 359 the request for full print job 309 is matched with the corresponding print job that is stored in the local spool system of the client 106. Thereafter, in block 363 the full print job 313 (FIG. 2) is transmitted to the print server 113 and in block 366 the full print job 313 is deleted in the client 106. Thereafter, the job send logic 159 reverts back to block 350.

With reference to FIG. 4, shown is a flowchart of the print queue logic 196 according to another embodiment of the present invention. The print queue logic 196 is executed by the processor 173 (FIG. 1) of the queue server 109 (FIG. 1) to perform the functions of the queue server 109 that relate to the printing protocol 300 (FIG. 2). Beginning with block 400, the queue server 109 determines whether it has received a

print job ticket 303 (FIG. 2) from the client 106 (FIG. 2). If so, then the print queue logic 196 moves to block 403. If not, then the print queue logic 196 moves to block 406. Assuming that the print queue logic 196 has moved to block 403, the print job ticket 303 that is received from the client 106 is placed within a printing queue maintained in the queue server 109. The printing queue may be stored, for example, in the memory 176 (FIG. 1). The print queue logic 196 then moves to block 406. Upon reaching block 406, the print queue logic 196 determines whether the printer 116 (FIG. 1) is either idle or printing a document. This is determined, for example, by transmitting the printer polling message 304 (FIG. 2) to the print server 113 to receive the printer response message 306. The printer response message 306 indicates whether the printer 116 is available. Alternatively, the print queue logic 196 may simply wait to receive a printer available signal from the print server 113. If the printer 116 is idle and ready to print a document in block 406, then the print queue logic 196 moves to block 409. Otherwise, the print queue logic 196 reverts back to block 400.

In block 409 the print queue logic 196 determines whether there is a job in the queue that is to be printed. If so, then the print queue logic 196 proceeds to block 413. Otherwise, the print queue logic 196 reverts back to block 400. In block 413, the print queue logic 196 transmits the print job ticket 303 (FIG. 2) to the print server 113. Then, in block 416 the print job ticket 303 is deleted from the printing queue maintained within the queue server 109 as the job has been sent to the print server 113 for printing. Thereafter, the print queue logic 196 reverts back to block 400.

With reference to FIG. 5, shown is a flowchart of the print server logic 236 that is executed by the processor 213 (FIG. 1) in the print server 113 (FIG. 1). The print server logic 236 is executed to perform the functions of the print server 113 that relate to the printing protocol 300 (FIG. 2). Beginning with block 450, the print server logic 236 determines if a printer polling message 304 (FIG. 2) has been received. If so, then the print server logic 236 moves to block 453. Otherwise, the print server logic 236 proceeds to block 456. In block 453 the print server logic 236 determines the print status of the print server 113 and the printer 116. When a print job is not being processed based on a previously received print job ticket 303, then the print status is available. Otherwise, the print status is unavailable. The actual print status may be determined in

block 453, for example, by examining a bit in the memory 216 (FIG. 1) that maintains the print status. Specifically, the bit may be placed in a logical "0" state indicating an available status or in a logical "1" state indicating an unavailable status. Based on the print status, the print server logic 236 sends an appropriate printer response message 306 (FIG. 2) to the queue server that originally sent the printer polling message 304. Thereafter, the print server logic 236 proceeds to block 456.

In block 456, the print server logic 236 determines whether a print job ticket 303 has been received from the queue server 109. If so, then the print server logic 236 moves to block 459. If not, then the print server logic 236 proceeds to block 463. In block 459 the print server logic 236 sets the print status to "unavailable". This may be accomplished, for example, by setting the previously mentioned printer status bit to a logical "1". Thereafter, in block 466 the network address of the client 106 (FIG. 1) from which the print job ticket 303 originated is determined. Next, in block 469 the print server logic 236 transmits a request for full print job 309 to the client 106. The print server logic 236 then proceeds to block 463.

In block 463 the print server logic 236 determines whether data from a full print job 313 (FIG. 2) is available from the client 106. If so, then the print server logic 236 responds by moving to block 473. If not, the print server logic 236 reverts back to block 450 as shown. In block 473, the data from the full print job 313 is applied to the printer 116 for printing and the print server logic 236 then moves on to block 476 as shown.

In block 476 the print server logic 236 determines whether the full print job 313 has finished printing on the printer 116. If so, then the print server logic 236 proceeds to block 479. If not, then the print server logic 236 reverts back to block 450 as shown. In block 479 the print server logic 236 sets the print status to "available" by setting the status bit previously mentioned, for example, to a logic "0". Thereafter, the print server logic 236 reverts back to block 450.

Although the logic 159 (FIG. 1), 196 (FIG. 1), and 236 (FIG. 1) of the present invention is embodied in software as discussed above, as an alternative the logic 159, 196, and 236 may also be embodied in hardware or a combination of software and hardware. If embodied in hardware, the logic 159, 196, and 236 can be implemented as a circuit or state machine that employs any one of or a combination of a number of technologies. These

technologies may include, but are not limited to, discret logic circuits having logic gates for implementing various logic functions upon an application of one or more data signals, application specific integrated circuits having appropriate logic gates, programmable gate arrays (PGA), field programmable gate arrays (FPGA), or other components, *etc.* Such technologies are generally well known by those skilled in the art and, consequently, are not described in detail herein.

The flow charts of FIGS. 3-5 show the architecture, functionality, and operation of an implementation of the logic 159, 196, and 236. If embodied in software, each block may represent a module, segment, or portion of code that comprises one or more executable instructions to implement the specified logical function(s). If embodied in hardware, each block may represent a circuit or a number of interconnected circuits to implement the specified logical function(s). Although the flow charts of FIGS. 3-5 show a specific order of execution, it is understood that the order of execution may differ from that which is depicted. For example, the order of execution of two or more blocks may be scrambled relative to the order shown. Also, two or more blocks shown in succession in FIGS. 3-5 may be executed concurrently or with partial concurrence. It is understood that all such variations are within the scope of the present invention.

Also, the logic 159, 196, and 236 can be embodied in any computer-readable medium for use by or in connection with an instruction execution system such as a computer/processor based system or other system that can fetch or obtain the logic from the computer-readable medium and execute the instructions contained therein. In the context of this document, a "computer-readable medium" can be any medium that can contain, store, or maintain the logic 159, 196, and 236 for use by or in connection with the instruction execution system. The computer readable medium can comprise any one of many physical media such as, for example, electronic, magnetic, optical, electromagnetic, infrared, or semiconductor media. More specific examples of a suitable computer-readable medium would include, but are not limited to, a portable magnetic computer diskette such as floppy diskettes or hard drives, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory, or a portable compact disc.

Many variations and modifications may be made to the above-described embodiment(s) of the invention without departing substantially from the spirit and

principles of the invention. All such modifications and variations are int nded to be included herein within the scope of the present invention.

CLAIMS FROM CO-PENDING US PATENT APPLICATION 9/640,855 AS AMENDED ON APRIL 8, 2004

1-21. (Canceled)

22. A method for printing on a network, comprising the steps of:

generating a print job ticket in a client, the print job ticket being associated with a full print job;

transmitting the print job ticket over the network to a queue server;

placing the print job ticket in a queue in the queue server;

determining in the queue server whether a printer is available to print the full print job associated with the print job ticket;

transmitting the print job ticket over the network from the queue server to a print server when it is determined that the printer is available to print the full print job, the print server being associated with the printer;

transmitting a request for a full print job from the print server to the client; and

transmitting the full print job from the client to the print server in response to the request.

23. The method of claim 22, wherein the step of transmitting a request for a full print job from the print server to the client further comprises the step of determining an address of the client on the network.

24. The method of claim 22, wherein the step of generating a print job ticket in a client further comprises the steps of:

writing a number of pages of the full print job into the print job ticket;

writing the address of the client into the print job ticket; and

writing a print job identifier into the print job ticket, the print job identifier being associated with the full print job.

25. A system on a network for printing a document, comprising:

a client coupl d to the n twork, the client having a first processor coupled to first local interface and a first memory coupled to the first local interface, the client further comprising send job logic stored on the first memory and executable by the first processor, the send job logic comprising:

logic to generate a print job ticket associated with a full print job; and

logic to transmit the print job ticket to a queue server;

the queue server being coupled to the network, the queue server having a second processor coupled to second local interface and a second memory coupled to the second local interface, the queue server further comprising print queue logic stored on the second memory and executable by the second processor, the print queue logic comprising:

logic to place the print job ticket in a queue in the queue server; and

logic to transmit the print job ticket from the queue server over the network to a print server; and

the print server being coupled to the network, the print server having a third processor coupled to third local interface and a third memory coupled to the third local interface, the print server further comprising print server logic stored on the third memory and executable by the third processor, the print server logic comprising:

logic to determine an address of the client on the network from the print job ticket received from the queue server;

logic to transmit a request for a full print job to the client; and

logic to apply the full print job received from the client to the printer.

26. The method of claim 22, wherein th step of determining in the queue server wh ther the printer is available to print the full print job associated with th print job ticket further comprises the step of transmitting a printer polling message from the queue server over the network to the print server to determine if the printer is available to print the full print job.

27. The method of claim 22, wherein the step of determining in the queue server whether the printer is available to print the full print job associated with the print job ticket further comprises the step of waiting in the queue server to receive a printer availability signal transmitted from the print server over the network to the queue server, the printer availability signal indicating that the printer is available to print the full print job.

ABSTRACT OF THE DISCLOSURE

A system and a method are provided for printing on a network. In one embodiment, broadly stated the method comprises the steps of: generating a print job ticket in a client, the print job ticket being associated with a full print job; transmitting the print job ticket to a queue server; placing the print job ticket in a queue in the queue server; transmitting the print job ticket from the queue server to a print server; transmitting a request for a full print job from the print server to the client; and, transmitting the full print job that encompasses a document to be printed from the client to the print server in response to the request.

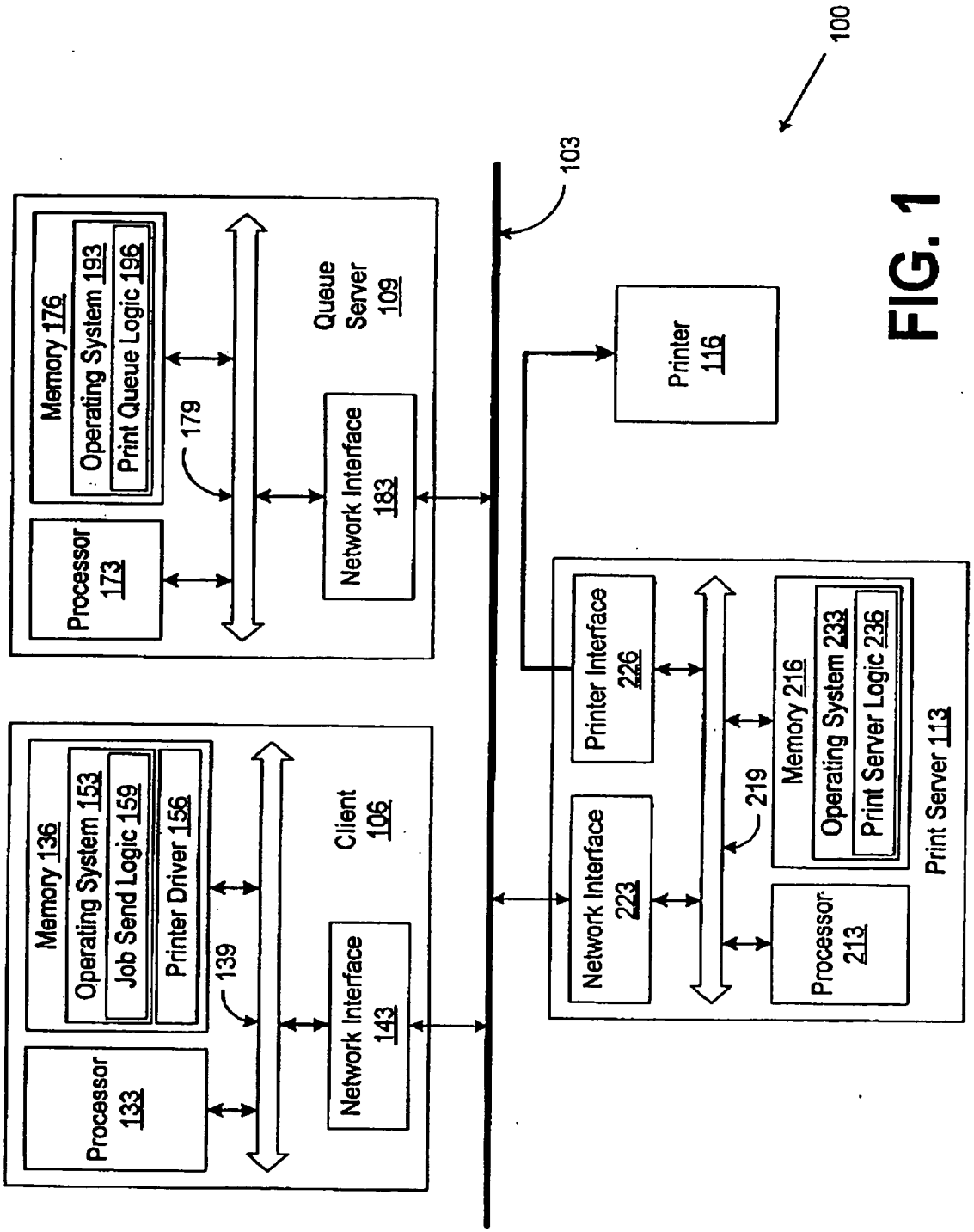

100
103
Queue Server 109
Memory 176
Operating System 193
Print Queue Logic 196
Processor 173
179
Network Interface 183
Printer 116
Client 106
Memory 136
Operating System 153
Job Send Logic 159
Printer Driver 156
Processor 133
139
Network Interface 143
Print Server 113
Printer Interface 226
Network Interface 223
219
Memory 216
Operating System 233
Print Server Logic 236
Processor 213

FIG. 1

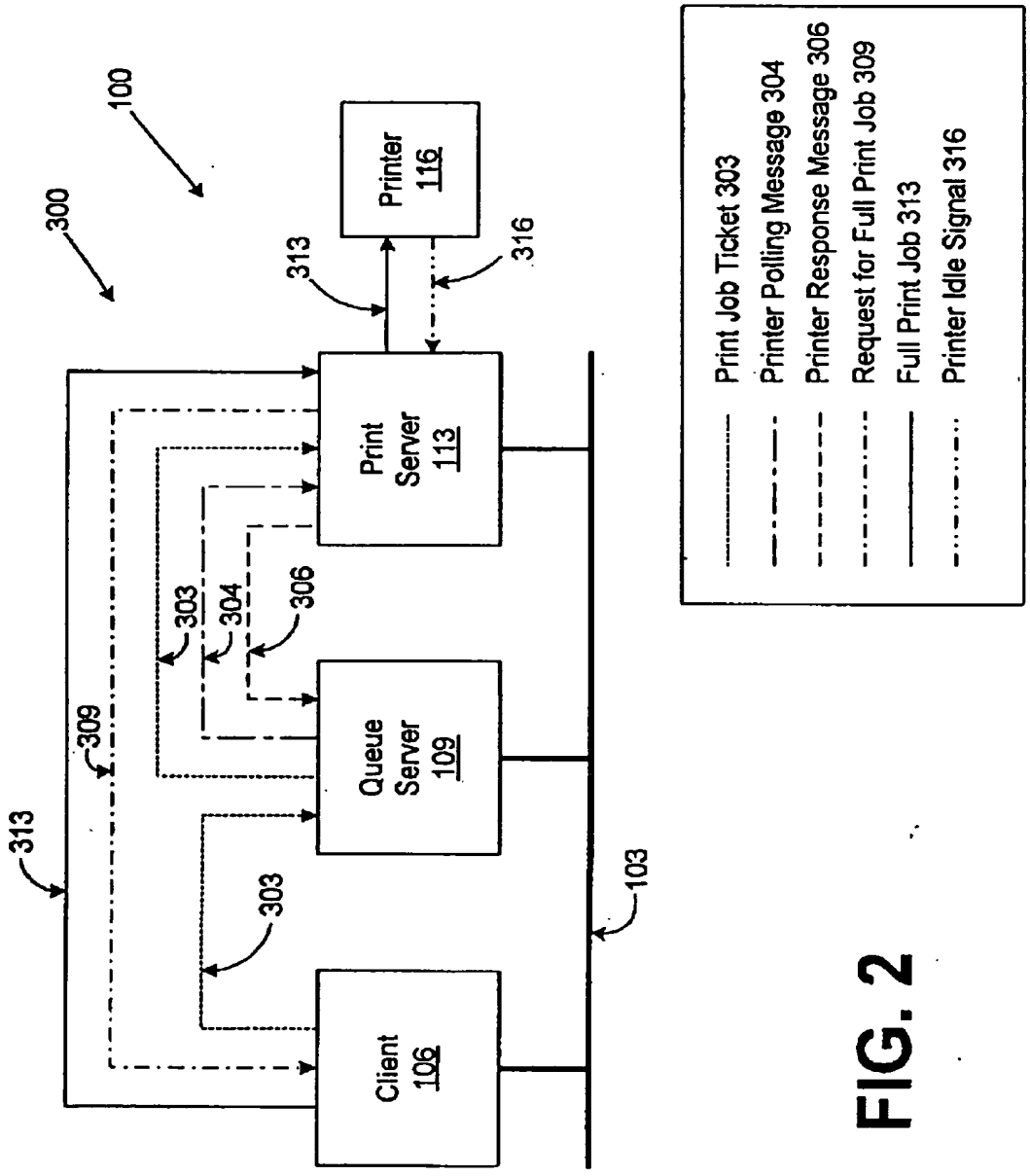

100
300
Printer 116
313
316
Print Server 113
303
304
306
309
Queue Server 109
313
303
103
Client 106
Print Job Ticket 303
Printer Polling Message 304
Printer Response Message 306
Request for Full Print Job 309
Full Print Job 313
Printer Idle Signal 316

FIG. 2

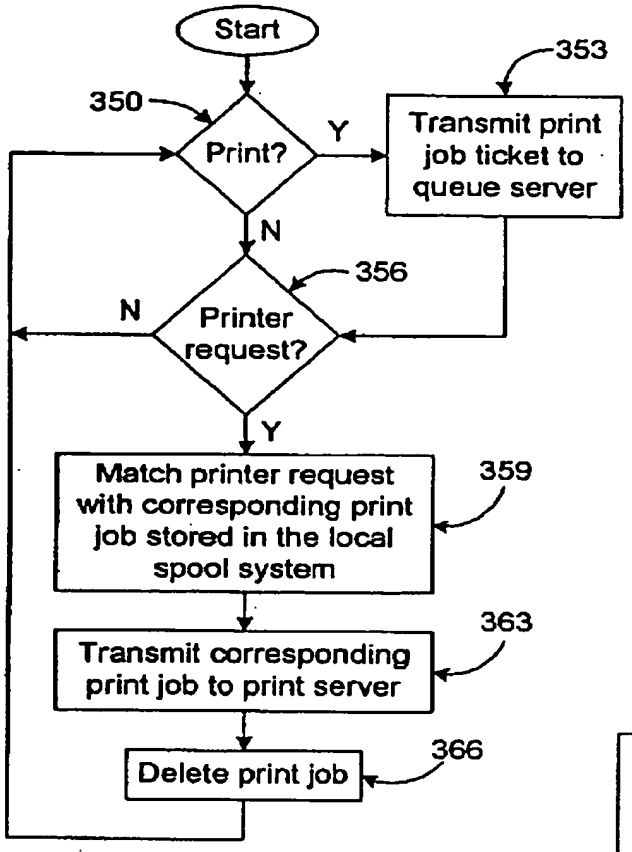
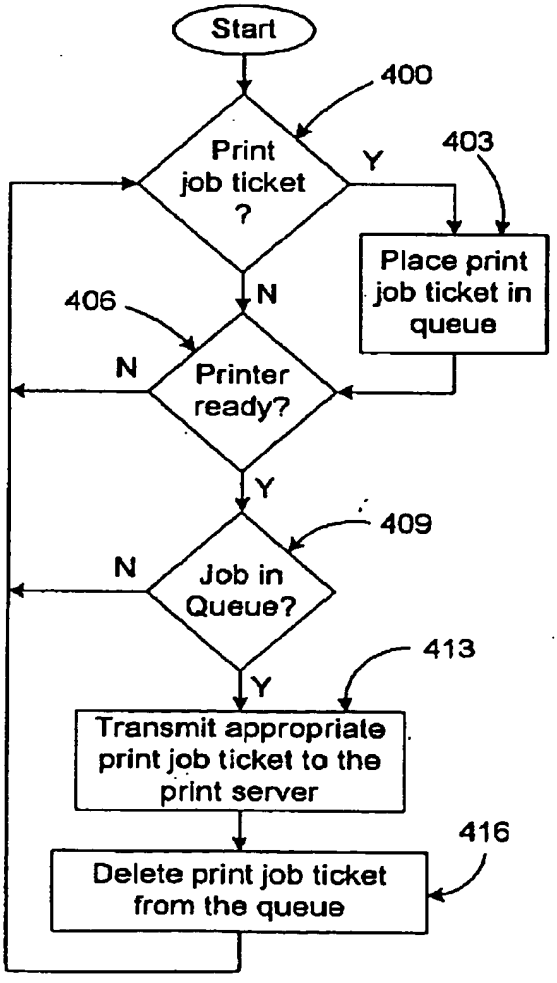

Start
350
Print?
Y
353
Transmit print job ticket to queue server
N
356
N
Printer request?
Y
Match printer request with corresponding print job stored in the local spool system
359
Transmit corresponding print job to print server
363
Delete print job
366
159
196

FIG. 3

FIG. 4


Start
400
Print job ticket ?
Y
403
Place print job ticket in queue
N
406
N
Printer ready?
Y
409
N
Job in Queue?
413
Y
Transmit appropriate print job ticket to the print server
416
Delete print job ticket from the queue

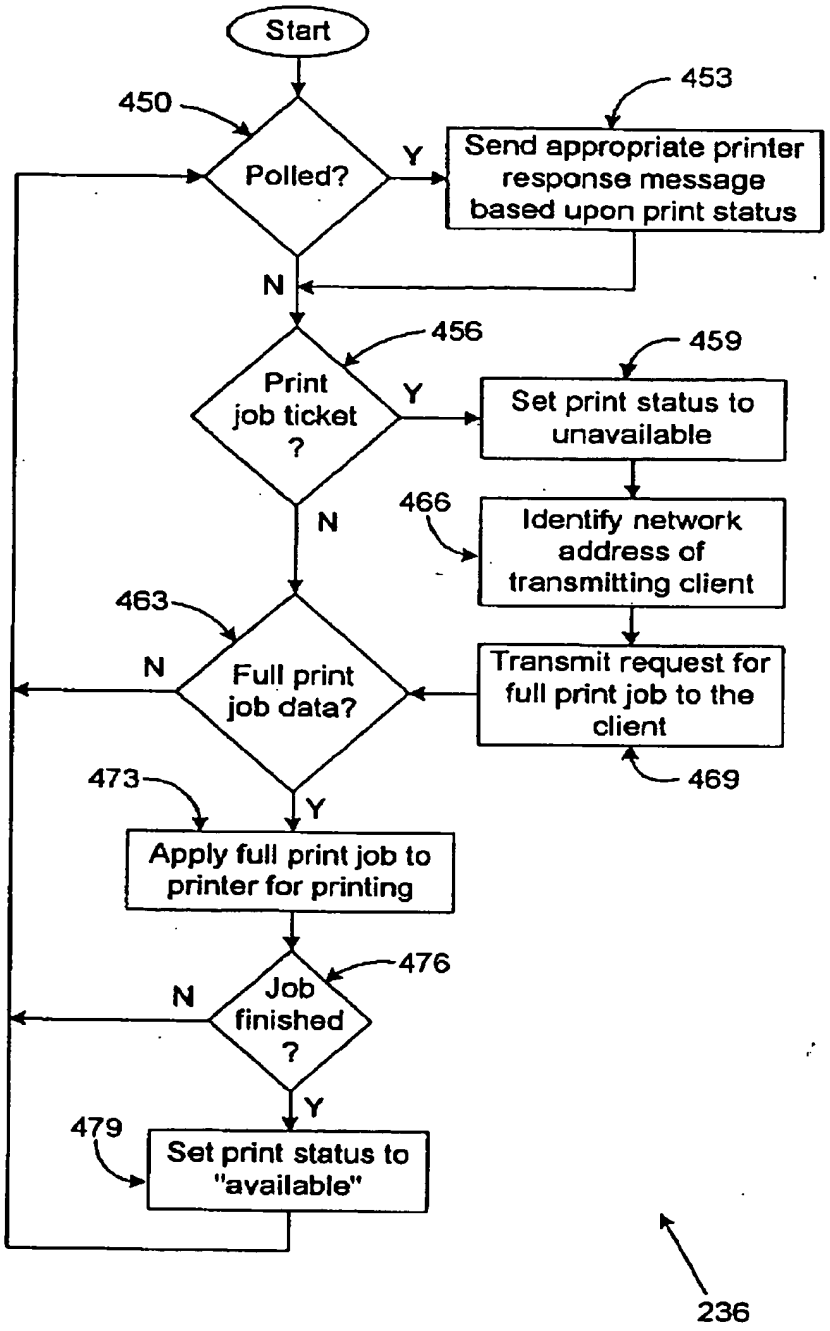

Start
450
Polled?
Y
453
Send appropriate printer response message based upon print status
N
456
Print job ticket ?
Y
459
Set print status to unavailable
N
466
Identify network address of transmitting client
463
Full print job data?
N
Transmit request for full print job to the client
469
473
Y
Apply full print job to printer for printing
476
Job finished ?
N
Y
479
Set print status to "available"
236

FIG. 5